Tutorial on Fourier Theory

Yerin Yoo

March 2001

1 Introduction: Why Fourier?

During the preparation of this tutorial, I found that almost all the textbooks on digital image processing have a section devoted to the Fourier Theory. Most of those describe some formulas and algorithms, but one can easily be lost in seemingly incomprehensible mathematics.

The basic idea behind all those horrible looking formulas is rather simple, even fascinating: *it is possible to form any function $f(x)$ as a summation of a series of sine and cosine terms of increasing frequency*. In other words, any space or time varying data can be transformed into a different domain called the *frequency space*. A fellow called Joseph Fourier first came up with the idea in the 19th century, and it was proven to be useful in various applications, mainly in signal processing.

1.1 Frequency Space

Let us talk about this *frequency space* before going any further into the details. The term frequency comes up a lot in physics, as some variation in time, describing the characteristics of some periodic motion or behavior. The term frequency that we talk about in computer vision usually is to do with variation in brightness or color across the image, i.e. it is a function of spatial coordinates, rather than time. Some books even call it *spatial frequency*.

For example, if an image represented in frequency space has *high* frequencies then it means that the image has sharp edges or details. Let's look at figure 1, which shows frequency graphs of 4 different images. If you have trouble interpreting the frequency graphs on the top low; The low frequency terms are on the

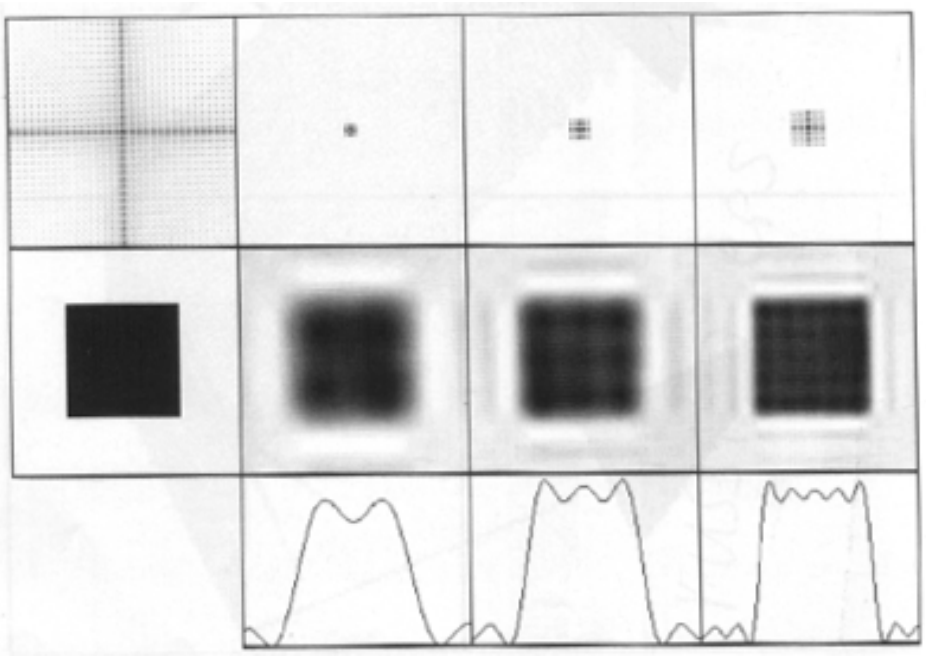

Figure 1: Images in the spatial domain are in the middle row, and their frequency space are shown on the top row. The bottom row shows the varying brightness of the horizontal line through the center of an image.
(Taken from p.178 of [1].)

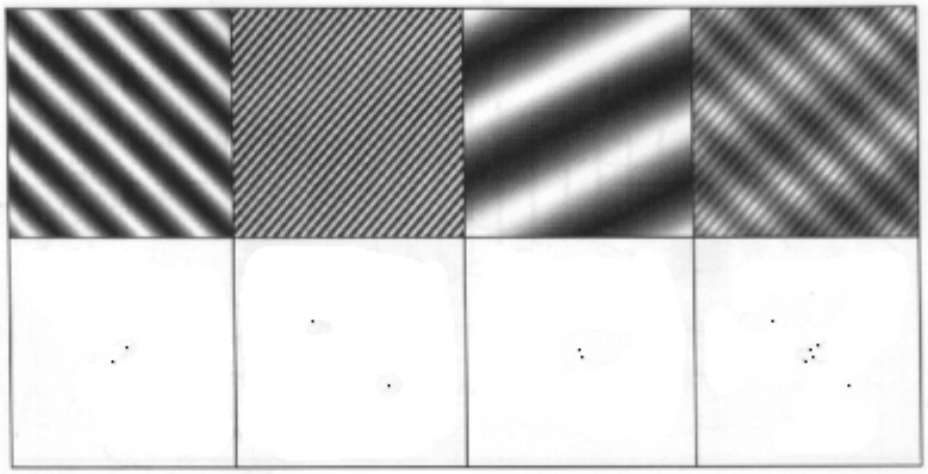

Figure 2: Images with perfectly sinusoidal variations in brightness: The first three images are represented by two dots. You can easily see that the position and orientation of those dots have something to do with what the original image looks like. The 4th image is the sum of the first three.
(Taken from p.177 of [1].)

center of the square, and terms with higher magnitude are on the outer edges. (Imagine an invisible axis with its origin at the center of the square.) Now, the frequency space on the top left consists of higher frequencies as well as low ones, so the original image has sharp edges. The second image from the left, however, is much fuzzier, and of course the frequency graph for it only has lower frequency terms.

Another thing to note is that if an image has perfectly *sinusoidal* variations in brightness, then it can be represented by very few dots on the frequency image as shown in figure 2. From those images, you can also see that regular images or images of repeating pattern generate fewer dots on the frequency graph, compared to images on figure 1 which don't have any repeating pattern.

1.2 So, What's the Point?

Frequency domain offers some attractive advantages for image processing. It makes large filtering operations much faster, and it collects information together in different ways that can sometimes separate signal from noise or allow measurements that would be very difficult in spatial domain. Furthermore, the Fourier

transform makes it easy to go forwards and backwards from the spacial domain to the frequency space.

For example, say we had an image with some periodic noise that we wanted to eliminate. (Just imagine a photocopied image with some dirty gray-ish spots in a regular pattern.) If we convert the image data into the frequency space, any periodic noise in the original image will show up as bright spots on the star-diagram[1]. If we "block out" those points and apply the inverse Fourier transform to get the original image, we can remove most of the noise and improve visibility of that image. (See figure 3 for the demonstration.)

More advantages of Fourier methods, and its applications will be discussed later in the tutorial.

2 Basics

Before really getting onto the main part of this tutorial, let us spend some time on mathematical basics. If you have sound background in mathematics, then you may skip this section and go to the next section.

2.1 Representing Complex Numbers

A complex number can be written as

$$R + iI,$$

where R and I are real numbers, and i is equal to $\sqrt{-1}$. R denotes a real part, and I denotes an imaginary part of a complex number. Real numbers can be thought of as the subset of complex numbers, where $I = 0$.

Example: $1 + 2i$, where $R = 1$ and $I = 2$.

Geometrically speaking, real numbers can fit on an infinitely long line in the 1 dimensional space. If we give one more dimension to it, then we can represent even more numbers, i.e. numbers that sit above and below the line of real numbers. Thus, complex numbers sit on a plane rather than a line. The horizontal axis of such representation is called the *real axis*, and the vertical axis the *imaginary axis*. Thus, a complex number $R + iI$ is coordinate (R, I) on this plane.

Example: $1 + 2i$ is on $(1, 2)$ of the complex number plane.

[1] A representation of an image data in frequency space.

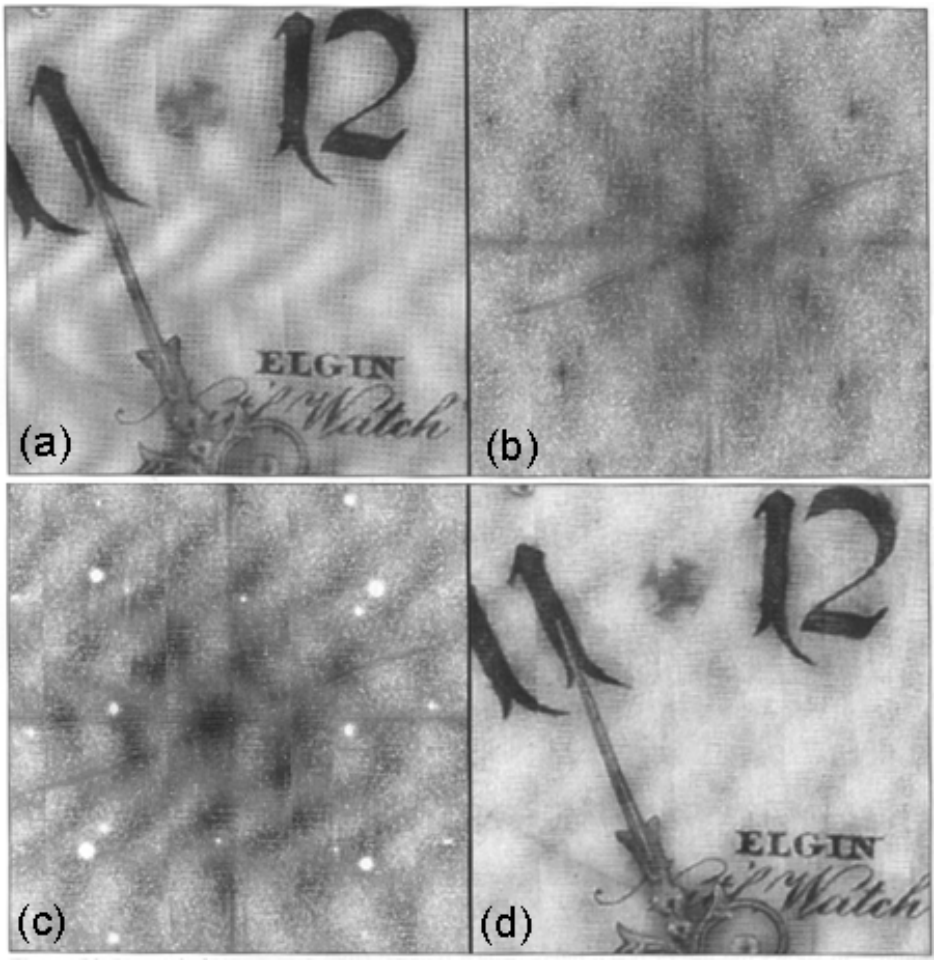

Figure 3: (a) Our dirty looking photocopied image. (b) The representation of our image in the frequency space, i.e. the star diagram. Look, you can see stars! (c) Those stars, however, do no good to the image, so we rub them out. (d) Reconstruct the image using (c) and those dirty spots on the original image are gone! (Images taken from p.204 of [1].)

The analogy of complex numbers being coordinates on a plane lets us represent a complex number in a different way. In the above paragraph, we talked about a complex number as a *rectangular coordinate*, but we can also write it as a *polar coordinate*, i.e. in terms of its distance from the origin (magnitude), and the angle that it makes with the positive real axis (angle):

$$r(\cos\theta + i\sin\theta),$$

where $r = \sqrt{R^2 + I^2}$, and $\theta = \tan^{-1}(\frac{I}{R})$. The fact that $\cos\theta = \frac{R}{r}$ and $\sin\theta = \frac{I}{r}$ makes it obvious that the two forms of representation denote the same complex number.

Example: $1 + 2i$ can also be expressed as $\sqrt{3}(\cos\theta + i\sin\theta)$, where $\tan\theta = \frac{2}{1} = 2$ which makes $\theta = 64.435$ degrees. The magnitude is $\sqrt{3}$ and the angle is 64.435 degrees or 1.107 radians.

2.2 Euler's Formula

Euler's formula is:

$$e^{i\theta} = \cos\theta + i\sin\theta, \tag{1}$$

where $e = 2.71828\cdots$, and θ is an angle which can be any real number. This is proven to be true for any real number θ.

This gives yet another representation of complex numbers to be:

$$re^{i\theta},$$

where r is the magnitude of a polar form of complex number, and θ is the angle.

Example: $1 + 2i$ can also be expressed as $\sqrt{3}e^{i\theta}$, where $\theta = 64.435$ degrees or 1.107 radians.

In some textbooks, a complex number is often expressed in the form of the Euler's formula without indicating so. (It is the case specially in the formulas associated with the Fourier transforms.) If you see anything in the form of $re^{i\theta}$, that be sure that you know that is is just an ordinary complex number. Furthermore, θ usually means the angle in *radians* if not indicated otherwise.

3 Fourier Transform

First, we briefly look at the Fourier transform in the purely mathematical point of view, i.e. we will talk about "continuous" or "infinite" things. I will assume that

you know what mathematical symbols like π, and e means, and you are familiar with the complex numbers. Beware that the upcoming section have complex mathematics in it, so if you suffer from "integral-o-phobia" then just skim through to the next section and look at the discrete Fourier transform. Remember that the Fourier transform of a function is a summation of sine and cosine terms of different frequency. The summation can, in theory, consist of an infinite number of sine and cosine terms.

3.1 Equations

Now, let $f(x)$ be a continuous function of a real variable x. The *Fourier transform of* $f(x)$ is defined by the equation:

$$F(u) = \int_{\infty}^{-\infty} f(x) e^{-i2\pi ux} dx, \tag{2}$$

where $i = \sqrt{-1}$ and u is often called the *frequency variable*. The summation of sines and cosines might not be apparent just by looking at the above equation, but applying Euler's equation (see Eq. 1 in the previous section) gives

$$F(u) = \int_{\infty}^{-\infty} f(x)(\cos 2\pi ux - i \sin 2\pi ux) dx. \tag{3}$$

Given $F(u)$, we can go backwards and get $f(x)$ by using *inverse* Fourier transform:

$$f(x) = \int_{\infty}^{-\infty} F(u) e^{i2\pi ux} du. \tag{4}$$

Equations 2 and 4 are called *Fourier transform pairs*, and they exist if $f(x)$ is continuous and integrable, and $f(u)$ is integrable. These conditions are usually satisfied in practice.

Note that the only difference between the forward and inverse Fourier transform is the sign above e, which makes it easy to go back and forth between spatial and frequency domains; it is one of the characteristics that make Fourier transform useful.

Some of you might ask what $F(u)$ is. $F(u)$'s are the data in the *frequency space* that we talked about in the first section. Even if we start with a real function $f(x)$ in spatial domain, we usually end up with complex values of $F(u)$. It is because a real number multiplied by a complex number gives a complex number,

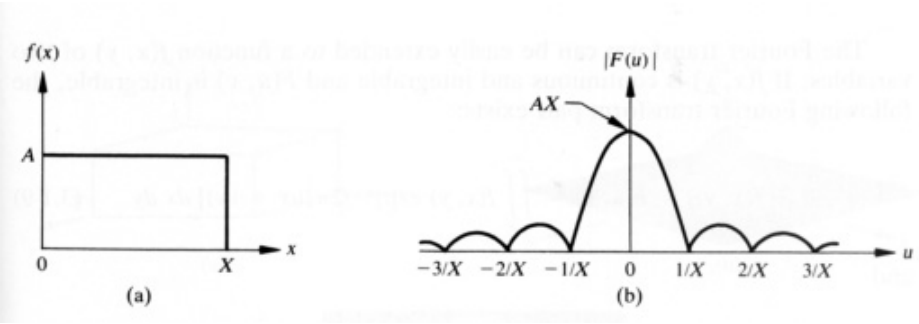

Figure 4: A simple function and its Fourier spectrum.
(Taken from p.83 of [2].)

so $f(x)e^{i2\pi ux}$ is complex, thus the sum of these terms must also give a complex number, i.e. $F(u)$. Therefore,

$$F(u) = R(u) + iI(u),$$

where $R(u)$ is a real component (terms that don't have i), and $I(u)$ is an imaginary component, (terms that involve i) when you expand the equation 3. (The "(u)" part is just there to remind you that the terms are the functions of u.) Just like any other complex numbers we can also write it in the polar form, giving $F(u) = r(\sin\theta + i\cos\theta) = re^{i\theta}$. In most of the textbooks, this form of the Fourier transform is written as

$$F(u) = |F(u)|e^{i\theta(u)}, \tag{5}$$

but it's essentially the same thing.

There are some words that we use frequently when talking about Fourier transform. The magnitude $|F(u)|$ from equation 5 is called the *Fourier spectrum* of $f(x)$ and $\theta(u)$ is *phase angle*. The square of the spectrum, $|F(u)|^2 = R^2(u) + I^2(u)$ is often denoted as $P(u)$ and is called the *power spectrum* of $f(x)$. The term *spectral density* is also commonly used to denote the power spectrum. The Fourier spectrum is often plotted against values of u. The Fourier spectrum is useful because it can be easily plotted against u on a piece of paper. (See figure 4 for an example of the Fourier spectrum.) Note that $F(u)$'s themselves are hard to plot against u on the 2-D plane because they are complex numbers.

3.2 Discrete Fourier Transform

Now that you know a thing or two about Fourier transform, we need to figure out a way to use it in practice. Going back to the example where we transform an image by taking brightness values from pixels, those pixel values are never continuous to begin with. (Remember that the Fourier transform we talked about in previous section was about a continuous function $f(x)$.) Our mathematicians came up with a good solution for this, namely *the discrete Fourier transform*.

Given N discrete samples of $f(x)$, sampled in uniform steps,

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-i2\pi ux/N} \tag{6}$$

for $u = 0, 1, 2, \ldots, N-1$, and

$$f(x) = \sum_{u=0}^{N-1} F(u) e^{i2\pi ux/N} \tag{7}$$

for $x = 0, 1, 2, \ldots, N-1$.

Notice that the integral is replaced by the summation, which is a simple "for loop" when programming. For those of you who are curious, the calculation inside Σ is multiplying $f(x) = R + Ii$ with $e^{i2\pi ux/N} = \cos(p) - i\sin(p)$ where $p = 2\pi ux$:

$$\begin{aligned} f(x) * e^{i2\pi ux/N} &= (R + Ii) * (\cos(p) - i\sin(p)) \\ &= R\cos(p) - Ri\sin(p) + Ii\cos(p) - Ii^2\sin(p) \\ &= R\cos(p) - Ri\sin(p) + Ii\cos(p) + I\sin(p) \\ &= (R\cos(p) + I\sin(p)) + i(I\cos(p) - R\sin(p)), \end{aligned}$$

where R, I are real numbers, and $I = 0$ when $f(x)$ is a real number.

The implementation of this transform in C is included in the appendix, but in the mean time, here is the pseudo-code in C style which I'm sure will make some readers happy:

```
/* Data type for N set of complex numbers */
double fx[N][2];
double Fu[N][2];

/* Fourier transform to get F(0)...F(N-1) */
```

```
for (u=0; u<N; u++) {
    for (k=0; k<N; k++) {
        p = 2*PI*u*k/N;
         /* real */
        Fu[u][0] += fx[k][0]*cos(p) + fx[k][1]*sin(p);
        /* imaginary */
        Fu[u][1] += fx[k][1]*cos(p) - fx[k][0]*sin(p);
    }
    /* multiply the result by 1/N */
    Fu[u][0]/=N;
    Fu[u][1]/=N;
}
```

3.3 Fast Fourier Transform

The discrete Fourier transform allows us to calculate the Fourier transform on a computer, but it is not so efficient. The number of complex multiplications and additions required to implement Eq. 6 and 7 is proportional to N^2. For every $F(u)$ that you calculate, you need to use all $f(0), \ldots, f(N-1)$ and there are N $F(u)$'s to calculate.

It turns out proper decomposition of Eq. 6 can make the number of multiplication and addition operations proportional to $N \log_2 N$. The decomposition procedure is called the *fast Fourier transform* (FFT) algorithm. There are number of different ways that this algorithm can be implemented, and we will not discuss this further in this tutorial. For those of you who are interested, there is a section devoted to this algorithm in "Numerical Recipes in C".

3.4 Applications of Fourier Transform

There are many situations in Graphics and Vision, specially in image processing and filtering, where Fourier transform is useful.

The Fourier method is often used on images from astronomy, microbiology, images of repetitive structures such as crystals and so on. It is because the Fourier transform is good for identifying a periodic component or lattice in an image. Identifying regular patterns on an image has other advantages like removing regular dirty spots or noise from an image as illustrated in figure 3 .

The example in figure 5 also shows another application of the Fourier transform; Some Fourier components of higher frequency can be removed to achieve

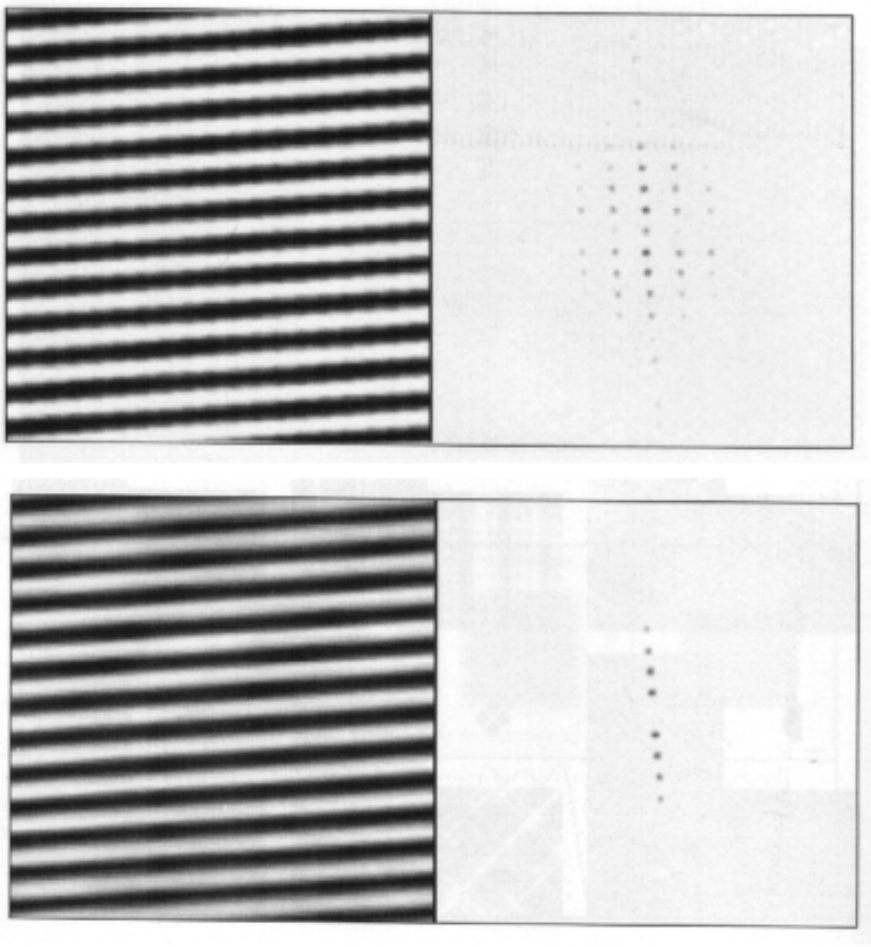

Figure 5: Top: Lines with zaggy edges, and the frequency components. Bottom: Removing some frequency components results in smoother lines.
(Taken from p.180-190 of [1].)

anti-aliasing effect, i.e. removing ugly zaggy edges.

There are other techniques associated with Fourier transform, namely convolution theory, correlation, sampling, reconstruction, image compression, and more. Since much of these topics were covered in Graphics course, I will include no further. Instead, the rest of the tutorial will focus on a particular application of the Fourier theory, namely the Fourier descriptors.

4 Fourier Descriptor

The *Fourier descriptor* is used to describe the boundary of a shape in 2 dimensional space using the Fourier methods.

4.1 Parameterization of the Boundary

First, we take N point digital boundary of a shape on xy-plane. We can choose to take all the pixels occupied by the boundary, or we can take N samples from them. This can be done by traveling the boundary anti-clockwise keeping the constant speed for, say, N seconds, taking a coordinate every second. [3] suggests a method for choosing an appropriate speed, which is out of the scope of this tutorial, so I will leave it to my keen readers to look it up.

Now we have a complete set of coordinates describing the boundary. We can call each coordinate (x_k, y_k) where $0 < k \leq N - 1$. You will have no trouble imagining these coordinates plotted on the xy-plane. Let us replace the labels on each axis; name the horizontal axis R for "real", and the vertical axis I for "imaginary". Now on the graph you have complex numbers that you know and love. We can call those $s(k)$'s,

$$s(k) = x_k + iy_k,$$

for $k = 0, 1, 2, \ldots, N - 1$. Although the interpretation of the sequence has been recast, the nature of the boundary itself has not been changed. The advantage of this representation is that it reduces a 2-D into a 1-D problem, i.e. you now have N complex numbers instead of 2*N real numbers.

4.2 Applying Fourier Transform

The discrete Fourier Transform of $s(k)$ gives

$$a(u) = \frac{1}{N} \sum_{k=0}^{N-1} s(k) e^{-i2\pi ux/N}, \tag{8}$$

for $u = 0, 1, 2, \ldots, N-1$. The complex coefficients $a(u)$ are called the *Fourier descriptors* of the boundary. Applying inverse Fourier Transform to $a(u)$ restores $s(k)$.

$$s(k) = \sum_{u=0}^{N-1} a(u) e^{i2\pi ux/N}, \tag{9}$$

for $k = 0, 1, 2, \ldots, N-1$. The restored pixel values are exactly the same as the ones that we started with.

However, we don't have to take all N pixel values to reconstruct the original image. We can "drop" the Fourier descriptors with higher frequencies because their contribution to the image is very small. Expressing this as an equation,

$$\hat{s}(k) = \sum_{u=0}^{M-1} a(u) e^{i2\pi ux/N}, \tag{10}$$

where $k = 0, 1, 2, \ldots, N-1$. This is equivalent to setting $a(u) = 0$ for all terms where $k > M-1$.

The more descriptors you use to reconstruct the original image, i.e. the bigger the M in the Eq. 10, the closer the result gets to the original image. (See figure 6.) In practice, we can reconstruct an image reasonably well even though we didn't use all the descriptors.

4.3 Geometrical Centroid

The descriptor $a(0)$ yields the geometrical *centroid* of the shape, with the x value given by the real part, and y value by the imaginary part. Consider only having the value of $a(0)$ as your Fourier descriptor, (and $a(1) = \cdots = a(N-1) = 0$) we can still try to reconstruct the original shape from it. The result we get is a circle, situated about the center of the original shape. This circle is called the *geometrical centroid*. In fact, eliminating all but the first 2 Fourier descriptors will always result in a circle. (Proof found in p456 of [4].)

The concept of the geometrical centroid relates perfectly well to the Fourier theory in general. When a function $f(x)$ is represented in the frequency space as

$F(u)$, $F(0)$ is the lowest frequency term which is the sinusoid making the major contribution to the image, i.e. the average brightness.

Talking more about the significance of this geometrical centroid, $a(0)$ is the only component in Fourier descriptors that is dependent on the actual location of the shape. For example, say, you had a shape which was centered on the origin, and you calculated Fourier descriptors of that image. Even if you translated the same image to somewhere else on the plane, you don't have to re-calculate the Fourier descriptors. All you need to adjust is the centroid, $a(0)$.

Moreover, the Fourier descriptors should be insensitive to other geometrical changes like rotation, scale and also the choice of the starting point. (The pixel point on the boundary where we start taking coordinates, i.e. (x_0, y_0).) It turns out that the Fourier descriptors are not strictly insensitive to these geometrical changes, but the changes can be related to simple transformations on the descriptors. (Table 8.1 on p.501 of [2] lists some basic properties of Fourier descriptors that might help you understand this point.)

4.4 Applications of Fourier Descriptors

Fourier descriptors are often used to smooth out fine details of a shape. As you have seen in the previous section, using the portion of Fourier descriptors to reconstruct an image smooths out the the sharp edges and fine details found in the original shape. Filtering an image with Fourier descriptors provides a simple technique of contour smoothing.

Fourier description of an edge is also used for template matching. Since all the Fourier descriptors except the first $a(0)$ do not depend on the location of the edge within the plane, this provides a convenient method of classifying objects using template matching of an object's contour. A set of Fourier descriptors is computed for a known object. Ignoring the first component of the descriptors, the other Fourier descriptors are compared against the Fourier descriptors of unknown objects. The known object, whose Fourier descriptors are the most similar to the unknown object's Fourier descriptors, is the object the unknown object is classified to.

Fourier descriptors can also be used for calculation of region area, location of centroid, and computation of second-order moments; Describing the specific techniques involving Fourier descriptors is out of the scope of this tutorial, but p.207 of [3] has various pointers to some journal articles for some further reading for those who are keen. Some limitations of Fourier descriptors are also discussed on the same page of [3], again giving various pointers.

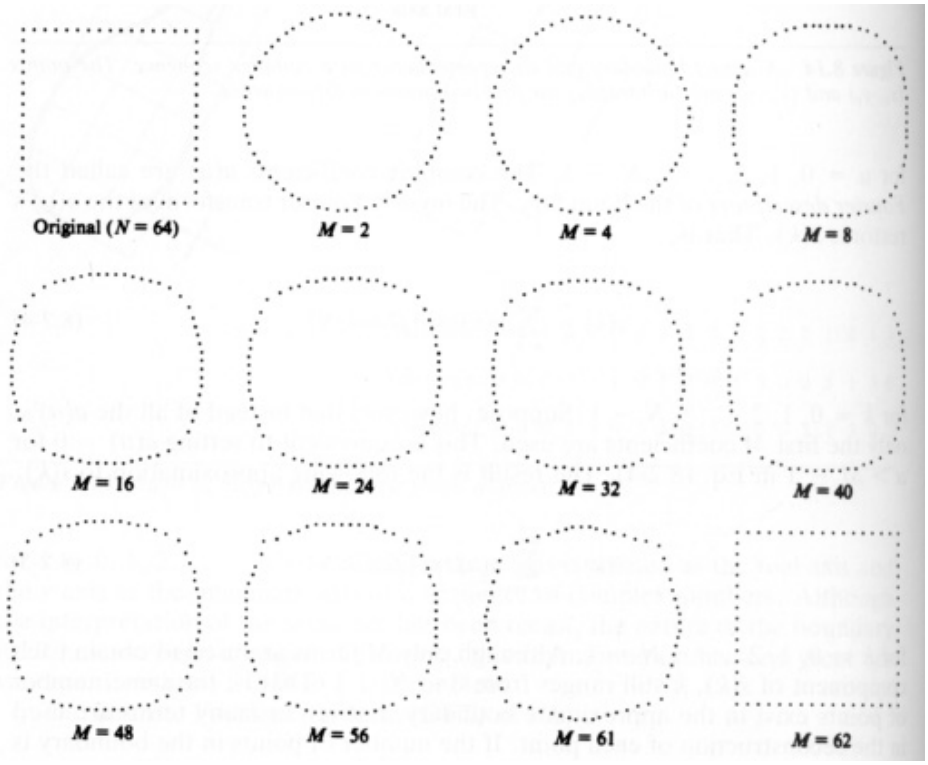

Figure 6: Examples of reconstructions from Fourier descriptors. M is the number of descriptors taken to reconstruct.
(Taken from p.500 of [2].)

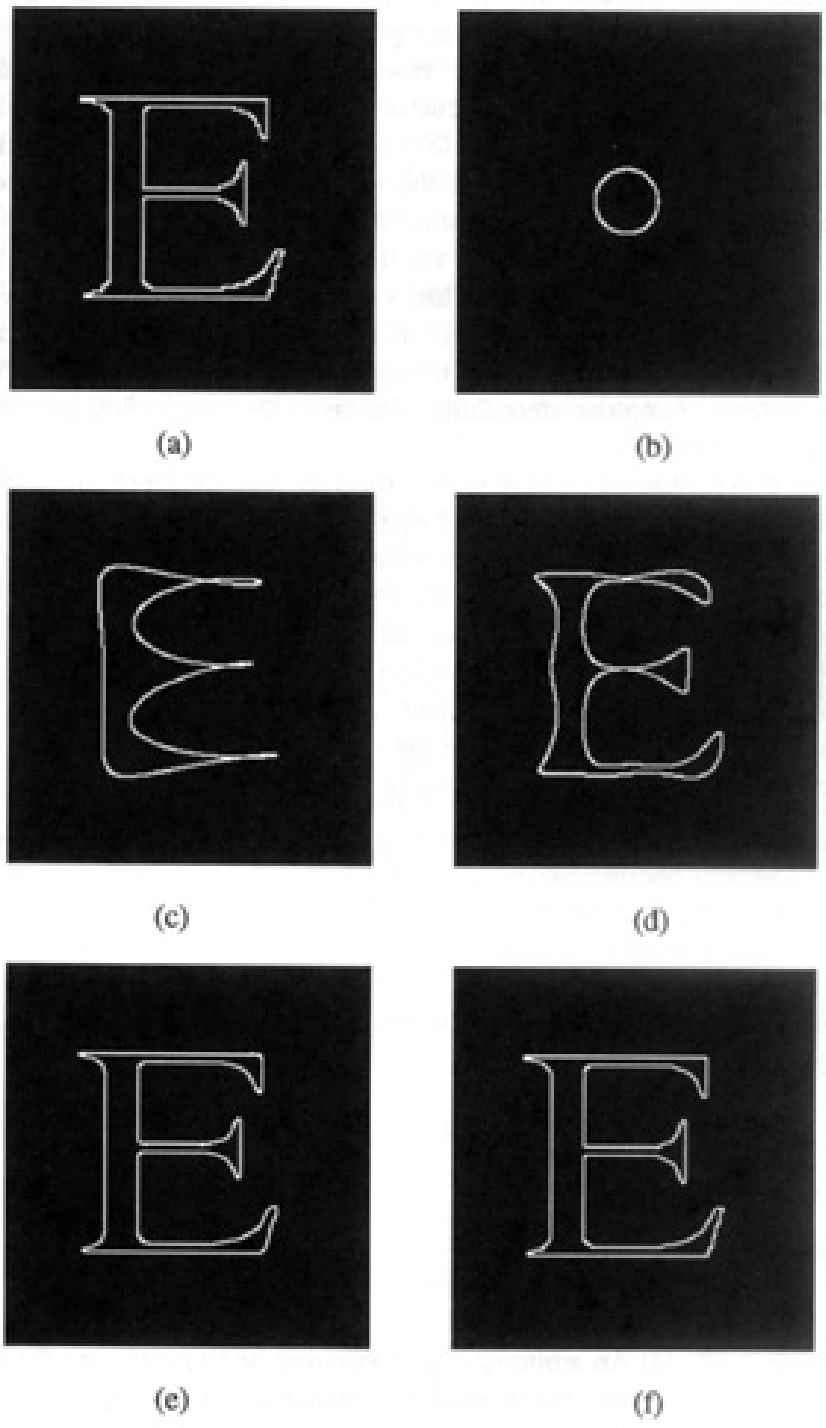

Figure 7: Examples using Fourier descriptors: (a) the original edge image with 1024 edge pixels, (b) 3 Fourier coefficients, (c) 21 Fourier coefficients, (d) 61 Fourier coefficients, (e) 201 Fourier coefficients, and (f) 401 Fourier coefficients. (Taken from p.457 of [4].)

5 Summary and Comments

The Fourier theory is based on the idea that any function can be composed of sines and cosines of different frequencies. In computer vision, images in the spatial domain can be transformed into the frequency domain by the Fourier transform.It is a very useful technique in image processing, because some operations and measurements are better be done in the frequency space than in the spatial domain. The implementation of the Fourier transform is called discrete Fourier transform (DFT), and other algorithms like fast Fourier transform (FFT) was also developed to reduce the complexity of the DFT. One of the techniques that involve the Fourier transform is the Fourier descriptors. They describe the boundary of a shape, and holds various properties that are useful in various applications.

The purpose of this tutorial is to give you the basic overview of the Fourier theory as well as to show some techniques that uses the Fourier theory. Like I said in the introduction, there are many textbooks that cover the Fourier theory, and I hope that you'll find those books less difficult to understand after going through this tutorial. Each textbooks that I looked at discuss somewhat different aspects of the Fourier theory, so here are some pointers to "where to look":

[2] seems to be the book that is referenced the most. It, however, has a humongous chapter describing all sorts of aspects of the Fourier transform without really explaining "why". The chapter is also very mathematically inclined, and it takes a while to see the relevance. I suggest that you look through the chapters when you feel somewhat confident about the topic.

[1] has very good introductory chapter on the Fourier theory. It's rather easy to read, and it has many good examples and figures.

[4] has a good section on the Fourier descriptors with good examples and figures. However, it is very basic, and the author never discusses the limitations of the technique.

[3] talks about the Fourier descriptors in 2 pages, and yet it seems to contain the most information. It is very hard to get the basic concepts, but it provides plenty of pointers to journal articles.

6 Focus Questions

Let us try out a few questions to make sure that we learned something:

1 Why is the Fourier transform useful in computer vision and graphics?

2 There are three ways of representing a complex number. Express the complex number, whose value of the real part is 3 and the imaginary part 4, by using those three representations.

3 You took 4 discrete samples of brightness values from a scan line of an image. Those were $f(0) = 0.5$, $f(1) = 0.75$, $f(2) = 1.0$ and $f(3) = 1.25$. Work out the first two data values in frequency domain by using the Fourier transform, i.e. $F(0)$ and $F(1)$. What are the Fourier spectrums of these terms?

4 What are the properties of the Fourier descriptor?

5 What are the advantages of using the Fourier descriptors to describe a shape?

References

[1] *The Image Processing Handbook*, chapter 4. CRC Press, 1992.

[2] *Digital Image Processing*, chapter 3 and 8.2.3. Addison-Wesley Publishing Company, 1993.

[3] *Image Processing, Analysis and Machine Vision*, chapter 6.2.3. Chapman and Hall, 1993.

[4] *Fundamentals of Electronic Image Processing*, chapter 8.4. IEEE Press, 1996.